Lab 10 - Intro to Recursion

Goals

By the end of this lab you should:

- know and be able to identify the base case(s) of a recursive function
- know and be able to identify the recursive step in a recursive function

Collaboration policy:

For this lab, collaboration IS ALLOWED.

Exercise 1: Recursive String Reversal

- For this exercise you will be using recursion to manipulate a string.
- Your function should then reverse the contents of the string using recursion.
- Your function may not use for loops, while loops, do while loops or any other iterative approach.
- The goal here is to implement a function that will have one string parameter that is to be passed by reference.
- The function definition should be named as follows: `void reverse(string &s);`
- Your file should be named `main.cpp`.
- It is important that you build a test harness to ensure your code is functioning correctly.
- **DO NOT USE GLOBAL VARIABLES.**

Exercise 2: Using recursion to display a list in reverse

Basic IntList Specifications:

You should use the IntList you developed for assignment 6. If you didn't complete it here are the minimum requirements for your class to pass this lab's harness.

You need a single header file (IntList.h) that declares and implements the IntNode class (just copy it exactly as it is below) as well as declares the IntList Class interface only. You also are required to come up with a separate implementation file (IntList.cpp) that implements the member functions of the IntList class. While developing your IntList class you must write your own test harness (main function). Never implement more than 1 or 2 member functions without fully testing them with your own test harness.

IntNode class

I am providing the IntNode class you must use. Place this class definition within the IntList.h file exactly as is. Make sure you place it above the definition of your IntList class. Notice that you will not code an implementation file for the IntNode class. The IntNode constructor has been defined inline (within the class declaration). Do not write any other functions for the IntNode class. Use as is.

```
struct IntNode
{
   int data;
   IntNode *next;
   IntNode( int data ) : data(data), next(0) {}
};
```

IntList class

Encapsulated (Private) Data Fields:

- head: IntNode *
- tail: IntNode *

Public Interface (Public Member Functions):

- IntList()
- ~IntList()
- void display() const
- void displayReverse() const;
- void push_front(int value)
- void pop_front()

Private Helper Functions

```
void displayReverse(IntNode *curr) const;
```

<u>Constructor and Destructor</u>

IntList() - the default constructor

Initialize an empty list.

~IntList()

This function should deallocate all remaining dynamically allocated memory (all remaining IntNodes).

Accessors

void display() const

This function displays to a single line all of the int values stored in the list, each separated by a space. It should NOT output a newline or space at the end.

void displayReverse() const

This function displays to a single line all of the int values stored in the list, each seperated by a space. It should NOT actually reverse the elements within IntList, it should only display the elements in reverse. This function MUST use recursion, and cannot use any for loops, while loops, do while loops or any other form of iterative methods. In fact, this function will just call the private helper function to display the list in reverse using recursion.

Mutator

void push_front(int value)

This function inserts a data value (within a new node) at the front end of the list.

void pop_front( )

This function removes the node at the front of the list (make sure you don't have a memory leak).

Private Helper Function

void displayReverse(IntNode *curr) const

This function is a helper function that will be used within displayReverse(). This function MUST use recursion, and cannot use any for loops, while loops, do while loops or any other form of iterative methods.

You **MAY NOT** define any other data fields, public or private, for this assignment.

IntList.h:

```
struct IntNode
{
   int data;
   IntNode *next;
   IntNode(int data) : data(data), next(0) {}
};
class IntList
{
  private:
   IntNode *head;
   IntNode *tail;
  public:
   IntList();
   void push_front(int value);
   void pop_front();
   void displayReverse() const;
  private:
   void displayReverse(IntNode *curr) const;
};
```

Compiling and Testing

You can compile the entire program, generating an executable named a.out, with the following command line:

```
g++ -W -Wall -Werror -ansi -pedantic main.cpp IntList.cpp
```

But when you want to compile the class seperately, without generating an executable, use the command line:

```
g++ -W -Wall -Werror -ansi -pedantic -c IntList.cpp
```

And then to generate the executable once you've already compiled the IntVector class separately, use the command line:

```
g++ -W -Wall -Werror -ansi -pedantic main.cpp IntList.o
```

What to Submit

For this lab you will turn in the following files (case sensitive) to R'Sub (galah.cs.ucr.edu):

- main.cpp (containing exercise 1's recursive function, not the IntList test harness)
- IntList.h
- IntList.cpp